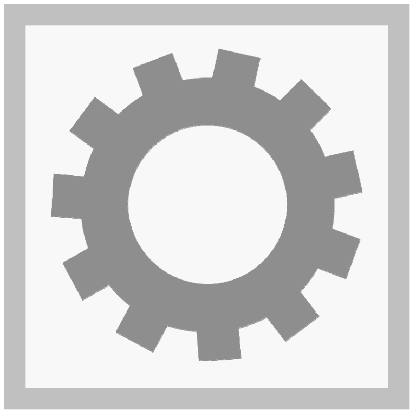
webMethods®

Get There Faster.™

The Definitive Guide to SOA Governance and Lifecycle Management

By Miko Matsumura, webMethods VP of SOA Product Marketing

March 2007



INTEGRATE. ASSEMBLE. OPTIMIZE.

TABLE OF CONTENTS

HOW TO AVOID READING THIS PAPER

Here's a word from the author: Busy people need a 45 page white paper like a hole in the head.

The paper is huge because it makes a substantial argument and supports that argument. Lucky for you, you can avoid reading the paper if you already *understand and agree with all ten of the statements below*. See how much time I just saved you? All you have to do is agree with everything I say. In general, this strategy is a great time-saver. I recommend that most people agree with everything I say.

If you don't understand or agree with all the statements, you may need to read the appropriate sections.

Quick Version in 10 Easy Steps:

1) **SOA adoption might be good**
2) **More groups adopting *might* be even better**
   — Challenges can also increase
3) **More groups adopt when…**
   — Lifecycle stakeholders join (like QA or IT deployment people)
   — Your company goes "Enterprise-scale SOA"
   — You go "B2B"
4) **But now you need SOA Lifecycle Governance!**
   — Because different groups have different concerns
5) **You can't buy SOA Governance "in a box"**
   — Because it's an art as well as a science
6) **But you might need ways to:**
   — Compose and store policies (Registry/Repository)
   — Enforce Design Time policy (Registry/Repository *again*)
   — Enforce Run Time policy (Message Transport)
   — Enforce Change Time policy (System Management Console)
7) **How about webMethods Infravio X-Registry ?**
   — It's based on a Registry/Repository
   — It has a lot of great features for end-to-end lifecycle governance (Design Time, Run Time and Change Time)
8) **If you need Run Time enforcement…**
   — We can work with most major Message Transports
   — Or you can also use webMethods Infravio X-Broker
9) **And if you need Change Time enforcement…**
   — We can work with most major System Management Consoles
   — Or you can also use NetIQ AppManager

10) **Therefore**

— X-Registry is the most complete SOA Governance and Lifecycle Management product available today

There, wasn’t that easy?

For people who want the complete story, by all means read on.

EXECUTIVE SUMMARY

"Prediction: In 2006, lack of working governance mechanisms in midsize-to-large (greater than 50 services) post-pilot SOA projects will be the most common reason for project failure (0.8 probability)."

Jess Thompson, Research Director, Gartner

This paper is designed to serve as "The Definitive Guide to SOA Governance and Lifecycle Management." Governance is the art and discipline of managing outcomes through structured relationships, procedures and policies.

Service Oriented Architecture (SOA) can transform how Information Technology (IT) is used by organizations. Benefits of SOA include: consolidation of IT systems, reusable services, faster standards-based integration, business flexibility, improved system visualization and greater degrees of compliance with business and regulatory policies.

However, without SOA Governance, organizations cannot achieve these benefits.

Although you can't buy SOA Governance in a boxed product, having working governance mechanisms in place, as Gartner suggests, can mean the difference between success and failure. Such systems can help automate complex governance processes, establish appropriate service lifecycles and help users visualize and secure their SOA. SOA has a lot of complex moving parts, and a good foundation for SOA Governance makes "doing the right things" easier and "doing the wrong things" harder.

This white paper describes just such a governance mechanism: The webMethods Infravio X-Registry 6 product. The X-Registry product is a Registry/Repository-based foundation for SOA Governance. This product is designed to support a distributed, federated SOA environment and to support interdependent stakeholders whether they are cross-lifecycle, cross-organization, or across companies and trading partners.

X-Registry consists of four governance applications:

- **Reuse Catalog** supports service reuse through advanced discovery, service artifact validation, interoperability testing, demo services, extended service profiles and service documentation.
- **Policy Center** provides dependency and impact analysis and configurable Service Delivery Contracts™, as well as detailed configuration of change time policy automation.
- **Administration Console** enables management of user roles and organizations. The Administration Console could also support a SOA dashboard that would provide key metrics and reports of services, performance and other analytics.

These applications help support the deployment of successful SOA projects. The product is built on a standards based core—supporting industry standards such as UDDI, LDAP, Java, JAXR, ebRIM, WSDL, WS-Policy, WS-Security and many others. These standards help assure that X-Registry supports a broad ecosystem of SOA Run Time environments, XML gateways, system management consoles and quality testing frameworks.

X-Registry is the highest rated SOA Governance Registry product reviewed by Infoworld Labs. X-Registry represents over 150 person-years of engineering and product development work, and is the most mature product of its type. The combination of powerful SOA Governance applications on top of a customer proven policy foundation make X-Registry the most complete SOA governance and lifecycle management platform available today.

This paper is designed to serve as "The Definitive Guide to SOA Governance and Lifecycle Management."

This white paper is divided into two sections.

- Part One addresses the topic of SOA Adoption and Governance. It does not assume extensive prior knowledge of the topic; users familiar with the topic are encouraged to skip ahead in Part One.
- Part Two introduces webMethods Infravio X-Registry, assumes familiarity with the content in Part One, and is designed for persons interested in evaluating X-Registry as a foundation for SOA Lifecycle Governance.

This paper is designed for Enterprise Architects, SOA practitioners, IT staff, software developers and informed lay persons with an interest in SOA.

Miko Matsumura is Vice President of SOA Product Marketing and Technology Standards at webMethods, Inc. He also serves as chair of the SOA Adoption Blueprints Technical Committee at Oasis and is the organizer of the SOA Link Interoperability Initiative. Miko regularly speaks throughout the world on SOA issues, as well as blogs at www.SOAcenter.com.

Prior to the recent acquisition of Infravio, Inc. by webMethods, Miko served as Vice President of Marketing and Technology Standards at Infravio, where he led marketing operations and strategic planning. Matsumura emerged as an industry thought leader at The Middleware Company, where he was a co-creator responsible for building the partner program for SOA Blueprints, the first complete vendor-neutral specification of an SOA application set, supported by BEA, Borland, HP, Microsoft, Oracle, Sun Microsystems, Veritas and others. At Systinet, Matsumura worked with the executive team and offshore development center on product development, product strategy, and outbound marketing, including representing the company at industry events. At Sun Microsystems, Matsumura held the position of Chief Java Evangelist, where he was a visible spokesperson for Java technologies and worked closely with Java ISVs and licensees to further the developer community. Before joining Sun, Matsumura worked at Wired Digital (acquired by Lycos) and the Well online community (acquired by Salon). He has also worked extensively with software start-up companies, including Biztone and

Kalepa Networks (acquired by Semio) raising more than 12 million in capital for Java start-ups.

Matsumura is currently a limited partner with Focus Ventures and was an advisor to the Asia Java Fund, as well as start-ups TogetherSoft (acquired by Borland), Dejima (acquired by Sybase) and Kendara (acquired by Excite). Matsumura holds an MBA from San Francisco State University and a Masters Degree in Neuroscience from Yale University.

PART ONE: SOA ADOPTION AND GOVERNANCE

WHAT IS SOA?

"Service Oriented Architecture is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations."

OASIS SOA Reference Model Technical Committee

Currently, the best standard definition for SOA is provided by the OASIS Service Oriented Architecture Reference Model. A reference model is an abstract framework for understanding significant relationships among the entities of some environment.

The SOA Reference Model contains definitions of commonly held terms designed to be appropriate to all implementations of SOA, regardless of technology. This white paper, unless specified otherwise will use terms such as Policy, Service, and Information Model as defined in the OASIS Reference Model, the latest versions which will be made available here:

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

SOA is most commonly referred to as an "Architectural Style." Therefore, projects that are implemented in this style bear many common characteristics.

Although the implementation details vary, the patterns common to SOA include:

- Interfaces and distributed capabilities
- Loose coupling
- Application composition
- Standards-based interoperability
- Mediation

Each of these topics will be reviewed below. Readers familiar with SOA patterns are encouraged to skip ahead to the next section, titled "SOA Adoption."

Key features of SOA include the use of remote service interfaces. The service interface separates the interface of a service from its implementation and allows the service to be used remotely. These service interfaces are usually collected in a form of lookup system known as a *Registry*.

Traditional software components are fused together in custom-fitted and proprietary way. This is referred to as "tight coupling." The opposite approach which allows consuming applications to flexibly route and bind to different service versions, providers, and implementations and to combine and recombine services is referred to as loose coupling.

The ability to create new services by combining existing deployed services is called service composition. Synchronous services are typically combined in portals, whereas long-lived asynchronous services are orchestrated into workflows and business processes.

Regardless of the underlying form (mainframe applications, object models, database schemas) information in SOA is typically expressed through standards.

In some cases SOA information such as configurations and messages are translated from native forms to XML-based standards to achieve interoperability. The machine readability of XML enables automated policy enforcement on both documents and messages.

Mediation is a special case of policy enforcement. Mediation policies enable the transformation, routing and securing of messages “mid flight” as a common pattern in SOA. Mediation is carried out by diverse systems such as Enterprise Service Bus, Brokers, Integration hubs, Adaptors and Gateways.

These patterns are common across a wide variety of technology implementations of SOA. In addition to the common patterns above, several other topics are also found in SOA—but are not considered fundamental or core SOA patterns including:

Event-Driven Architecture is a variation on the theme of SOA. An example of this technique is Staged Event Driven Architecture or SEDA, which is a way to support distributed state machines. It is also handy for breaking up problems that have been traditionally impossible to implement transactionally, due to the overall scope of the transactions (too big, too long). Event Driven Architecture has been linked with areas such as agent-based systems and takes advantage of interoperable services supplied by SOA.

Increasingly, SOA is seen associated with Business Process Management (BPM). This is because the interoperable services supplied by an SOA can be sequenced by a choreography language such as Business Process Execution Language (BPEL).

SOA ADOPTION

Service Oriented Architecture projects use patterns such as those listed above. However, unique requirements and patterns emerge in post-pilot SOA projects. Typically after the pilot, SOA projects tend to scale across organizational boundaries and involve additional stakeholders.

This paper uses the term “SOA Adoption” to refer to the manner in which SOA project grow within organizations. This growth can be experienced in terms of the number and variety of stakeholders who can benefit from the SOA.

These stakeholders can include the following:

In an SOA pilot, all of the complexity of SOA is constrained to a single organization. Even if the SOA project is constrained to a single business unit, it can begin to take on the characteristics of Enterprise SOA when multiple stakeholder groups get involved in the service lifecycle. In this case, multiple stages such as development, quality testing, staging and deployment (production) can emerge.

Finally, a full-scale Enterprise SOA project can cross boundaries between business units, between companies, and bridge the gap between central Enterprise-wide Information Technology (IT) services and the IT services at the business unit level.

Other ways that an SOA project can cross organizational boundaries are SOA projects that involve customers, suppliers and partners. By involving external organizations, this provides a need for additional governance mechanisms to manage the obligations and requirements of managing those relationships.

The emphasis in this paper is on understanding the unique opportunities and challenges of building out SOA systems beyond the pilot stage, particularly the governance challenges that grow out of interdependent SOA systems that involve multiple stakeholder organizations.

POTENTIAL BENEFITS OF SOA ADOPTION

"Step Five: Choose and Deploy a Registry or Repository."
" Step Six: Start Tackling Governance."

From "Ten Steps to SOA, Infoworld Special Feature"
Eric Knorr, Editor, Infoworld

SOA has the *potential* to provide many benefits. These benefits are related to controlling cost and complexity but also enabling an organization to evolve and compete with increased agility. As an SOA grows to Enterprise-scale, both the risks and opportunities also expand.

At Enterprise scale, breaking down the barriers between traditional IT "silos," SOA architects can create benefits for central IT, business units, and the organization as a whole.

Central IT can control cost, complexity and risk through:

- Consolidation
- Reuse
- Compliance

Business units can pursue opportunities by increasing:

- Agility and Flexibility
- Business Visibility
- Process Integration

SOA brings these two organizations closer together through:

- Business/IT Alignment

Each of these potential benefits is described below in more detail.

SOA provides organizations with opportunities to consolidate systems. Whether this is achieved through standardizing service interfaces and replacing (or outsourcing) implementations or through utility-style provisioning of shared server resources, SOA provides ample opportunity for an organization to move from a “siloed” model into a shared resource model, thereby increasing efficiencies of utilization.

In SOA, reuse of services can dramatically shorten the time to develop new application functionality. Service reuse is substantially unlike reuse in previous generations of software (e.g. “Object-Oriented Programming”) for several reasons. Firstly, the reuse happens at the level of fully instantiated and provisioned services (as opposed to source code). Secondly, SOA reuse is typically coarsely grained, which means that much larger “chunks” of software are able to be reused in each case.

Intermediary enforcement can also be configured to enable new compliance applications including auditing and reporting as well as security and business policies such as Service Level Agreements.

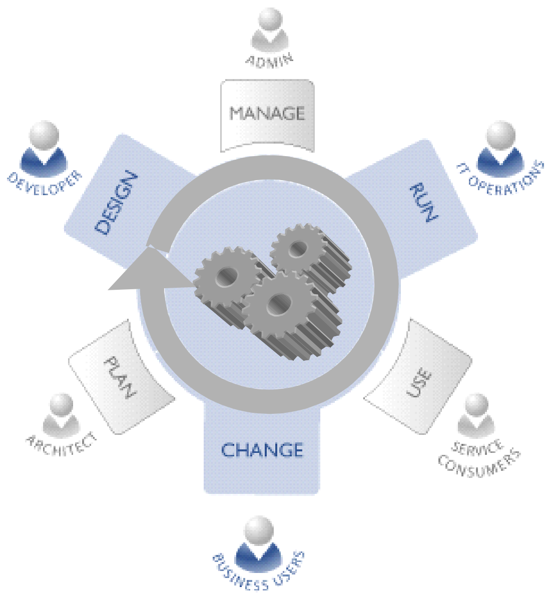
By “externalizing” rapidly changing elements of the IT system into metadata, SOA provides an opportunity to dramatically increase the speed of IT change. Whether that metadata is business processes encoded as BPEL (Business Process Execution Language) files, or Run Time Service Level Agreements (SLA), the ability to transform the behavior of deployed IT systems through metadata without touching a line of code provides an agility favored by business stakeholders.

SOA based on standards enables a new layer of intermediary-enabled transparent messaging. Enhanced visibility in the network allows a new ability to analyze the significance of IT system behavior. This enables a stronger connection between technology systems and business results which is sometimes referred to as Business Activity Monitoring (BAM).

The standards-based communications in SOA enables disparate IT systems to seamlessly and rapidly integrate. This applies to multiple divisions consolidated through acquisition or to networks of trading partners. These disparate systems can be coordinated by higher-level descriptions that focus on the state transitions between lower-level services.

Coarse-grained business services have interfaces that are easily understandable by business users. An architecture that assembles a set of such interfaces can create alignment between business users, who understand what the value of the IT services are, and technical users who understand the details of how to implement those services.

As SOA deployment matures, the number of stakeholders across its lifecycle increase. The diagram shows a simple service lifecycle including the planning, development, management, operation, consumption, and modification of services and policies.



By organizing Information Technology around the development, delivery and support of services and policies, a broader lifecycle of stakeholders can be aligned and the full costs and benefits of IT programs can be understood. From this perspective, Service Orientation can be seen as an extension of IT Governance best practices.

In addition to aligning stakeholders across the service lifecycle, a Governed SOA environment can align organizations such as centralized Information Technology and Lines of Business. Governed SOA can also be used to align organizations with external suppliers, customers and partners.

SOA ADOPTION CHALLENGES

"A platform that does not leverage shared registry and repository capabilities, cross-technology metadata management, and end-to-end service-level monitoring can result in not only tremendous waste because of redundancy and extra transference steps but also increased risk to the semantic integrity of the environment."

Sandra Rogers, Research Director, IDC

SOA provides unique opportunities, but also brings with it a new set of challenges, particularly as implementations grow to Enterprise scale. The complexity of integrating acquisitions or reconciling business unit operations are just a part of the challenges of SOA at large scales.

The primary challenges of SOA Adoption involves ensuring that stakeholders are able to meet their obligations to each other without compromising the requirements of the whole organization.

Most Enterprises can justify SOA by retiring redundant and underutilized capabilities all over their organization. However, most redundancy in Enterprises evolved for organizational and political reasons rather than technical reasons. People in organizations tend to hoard resources. They tend to be reluctant to lose control over the infrastructure that supports their business.

This means that redundant functionality is frequently found across organizational boundaries within application “silos.”

Organizational boundaries, resource issues and the complexity of incorporating requirements from disparate stakeholders can result in a lack of trust across the enterprise.

In order to achieve consolidation and reuse, an organization will require policies that govern the use, reuse and abuse of services. Just bringing down the walls that separate application “silos” and creating a free-for-all is a recipe for chaos. Thus business units will have to follow new sets of rules in order to participate in the shared resource system. These additional rules and procedures cost organizations both time and money. Reuse needs to be supported by ensuring that services comply with interoperability standards. Ensuring compliance with interoperability standards creates a cost.

Once reuse is established, new challenges emerge. By calling remote services which may be outside of the control of your organization, spaghetti-like chains of dependency can be formed which can result in unexplained cascading outages and mysterious spikes in usage. These system behaviors can result in lost opportunities or the requirement for wasted provisioning of overcapacity.

Systems with complex chains of dependencies can fail in equally complex ways. This means that the system can fail in ways that may be difficult to interpret and assign accountability. This can result in complex networks of blaming and finger-pointing.

Enterprise SOA requires business units to give up control of key resources, comply with centrally dictated standards and practices and subject their activities to centralized oversight and scrutiny. This does not sound like fun.

The organizational costs of complying with Enterprise policies and of becoming dependent on service functions outside a business unit's control (and hence more vulnerable to outages) are only attractive if the SOA can offer significant advantage in agility and business visibility.

Balancing these extra costs, risks and benefits requires new organizational behaviors and new rules and, in many cases, buy-in from high levels in the organization. By creating a redistribution of costs and benefits, an SOA project can ensure that stakeholders are motivated to participate and offer one another the benefits of agility and reuse.

The organizational balancing act needed to reorganize IT infrastructure involves a new set of incentives to promote reuse of services. Without changing the organizational and motivational structures, the same behaviors that created application "silos" are likely to continue.

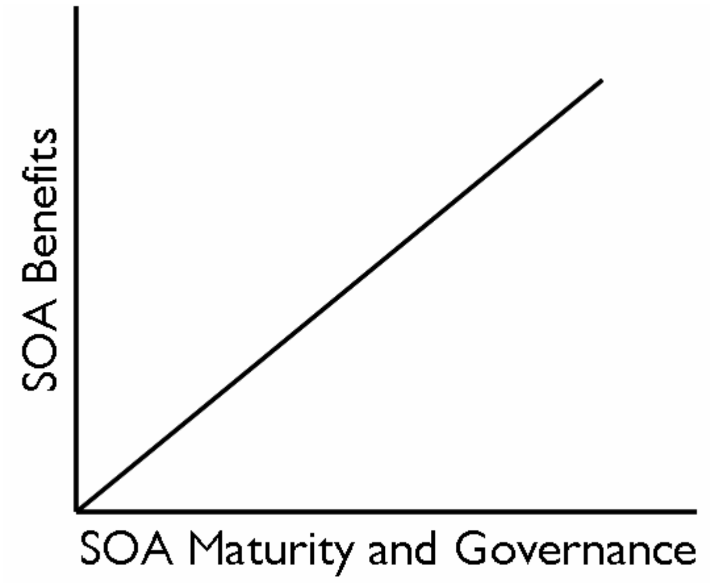
The new organizational behaviors required are part of the "art" of SOA Governance and cannot be bought as a packaged product. However, the new rules and enforcement mechanisms can be automated using foundational governance systems.

SOA GOVERNANCE

“SOA is a mess waiting to happen. By encouraging widespread reuse of scattered software components, SOA threatens to transform the enterprise network into a complex, sprawling, unmanageable mesh. Left ungoverned, SOA could allow anyone anywhere to deploy a new service any time they wish, and anyone anywhere to invoke and orchestrate that service—and thousands of others—into ever more convoluted messaging patterns. In such an environment, coordinated application planning and optimization become fiendishly difficult. In addition, rogue services could spring up everywhere and pass themselves off as legitimate nodes, wreaking havoc on the delicate trust that underlies production SOA.”

James Kobielus, principal analyst at Current Analysis

Most organizations entering mature SOA Adoption are hearing about the need for SOA Governance.



This paper holds as axiomatic that the benefits that organizations derive from SOA post-pilot are proportional to the maturity of the architecture and the strength of the Governance capabilities.

Some degree of benefit from using SOA patterns can be obtained without strict governance for example, the use of point-to-point Web Services for integration. However, point-to-point Web Services integration can hardly be called “SOA,” and any attempt to scale up such an “architecture” will most certainly fail.

In order to proceed with a course of action around SOA Adoption and Governance, a clearer definition of SOA Governance is needed. This paper refers to SOA Governance in the following context:

Governance is the art and discipline of managing outcomes through structured relationships, procedures, and policies.

SOA Governance is essentially the same as Governance, described above. However, SOA in particular requires unique sets of relationships, procedures and policies.

Since SOA is used in the design of software systems, special care must be applied to the creation, communication, enforcement, maintenance and adaptation of machine enforceable policies across the SOA lifecycle of Design Time, Run Time and Change Time.

As with other forms of Governance, relationships, policies and processes among organizations and human participants are also of great significance.

It's no secret that SOA has too many moving parts. This means that without mechanisms of control and enforcement, business policies can be breached (resulting in individuals acting in ways that hurt the organization) and technical policies can be breached (resulting in nonfunctioning, inefficient or noncompliant technical services).

Policies are guidelines that influence decisions or actions. The actions can be carried out by humans or machines. Policies typically have a "target set" or "filter" which determines when they are in force. For example, an equal opportunity policy applies to the set of employment applicants to the company. An executive retention policy applies to the set of managers ranked above "E10" in the HR system. A security policy can be applied to "all messages passed outside the firewall."

In the next section, we will see how SOA Governance can be planned around relationships, policies and processes across the SOA lifecycle.

CREATING AN SOA GOVERNANCE BLUEPRINT

SOA reuse should not resemble children grabbing piñata candy. Properly governed SOA should resemble the orderly design and construction of systems or structures for use by organizations. As such, the design and construction of buildings is an appropriate way to describe SOA implementation. For buildings, the architectural integrity is determined by:

1. The construction materials
2. The blueprint
3. The foundation

By following this metaphor, we can develop a best practices methodology for ensuring the delivery of SOA benefits.

Governance planning should be proportional to the scale and maturity of the SOA, much like a skyscraper requires better materials, better planning and a stronger foundation than a single-family house. The effort of SOA planning should be proportional to the anticipated benefits. Therefore it's important to understand the eventual scope and goals of the project from its outset.

A good SOA inventory begins as a laundry list but ends up more like a grocery list—it's important to list everything you have, but towards the ultimate goal of simplifying and rationalizing your catalog.

For SOA, the construction materials consist of existing IT systems and services. Taking an inventory of existing services is an important first step in creating an SOA plan. In addition to noting existing software services, take an inventory of the organizations that support and develop these assets. These are the building blocks which will be used to assemble your SOA.

A typical "bottom up" approach to creating an inventory of raw materials in your SOA is to spread the word that you are developing a catalog of available services. You may be surprised by the number and diversity of available services. In Enterprise SOA, redundant functions are most frequently found in application "silos" which cross organizational boundaries.

Instead of throwing all services into a massive catalog, SOA best practices suggest an intentional approach and a focus on assembling a core set of reusable services that represent coarse-grained decomposition of key business services provided by your organization. Another "top down" approach involves analyzing your business in terms of business activities and processes. These processes can then be decomposed into a sequence of steps, each of which should be supported by a business service.

For more information on process centric service decomposition, please refer to the OASIS SOA Adoption Blueprints, which provides a methodology and notation for this practice, available at the following URL:

http://www.oasis-open.org/committees/download.php/15071/A%20methodology%20for%20Service%20Architectures%201%202%204%20-%20OASIS%20Contribution.pdf

You may find redundancy or, in some cases, important capabilities that are not provided. This process is essential for developing an architectural view of your assets and understanding where to focus resources.

The Blueprint sets out the most important principles of the SOA—the responsibilities and expectations of each constituent part to each other and to the whole.

Without an appropriate blueprint and SOA, even the best services can degenerate into distributed "spaghetti" and mutually codependent nodes which can all be brought down by the weakest connection. A good blueprint begins by establishing organizational goals. Some typical goals are associated with the list of benefits.

Central IT Goals

- Consolidation
- Reuse
- Compliance

Business Unit Goals

- Agility and Flexibility
- Business Visibility
- Process Integration

Shared Goals

- Business/IT Alignment

Typically it is the job of an Enterprise Architect to develop the SOA Blueprint. However, for Enterprise-scale SOA, one best practice is to establish an SOA Competency Center, Center of Excellence, Architecture Council or other cross-functional group. Any SOA involving cross-lifecycle and cross-organizational stakeholders should consider a governance organization which includes representation from each group. This group is frequently used to establish governance roles, processes and policies, and is a key approval authority for any blueprint documents.

A good blueprint will document and establish polices regarding the following:

- Metrics, reporting and auditing
- Interoperability standards
- Service lifecycle processes
- Security policies

Policies defines how resources are marshaled under different patterns of circumstances; in short, *what* to do *when*. Of course in some cases it defines what not to do. Policy enforcement can be carried out by people (for example, approval policies) or by machines (for example, encryption policies). Each of these areas are addressed below:

Once the desired benefits are documented, these need to be further codified into success metrics. These measurable outcomes should be associated with each of the goals of the project. The polices about what to measure result in key performance indicators (KPIs) used to govern and optimize your SOA. Related to this might be policies about auditing and reporting, which should also be documented in the blueprint.

Another set of policies needed in an SOA Blueprint are the set of interoperability standards supported by the organization. Many of the benefits such as reuse and flexibility can be compromised by systems that can't interoperate.

Another set of policies established by the blueprint are service lifecycle processes. Best practices involve ensuring that appropriate stakeholders are involved in the migration of services throughout their lifecycle.

Most importantly, policies around security, authorization, access control and permissions should be established.

Despite the need for planning, it's important to see SOA as driven by iteration and successive optimization. Otherwise it can be easy to be trapped in endless planning and meetings and the kind of bureaucracy that is anathema to SOA agility. Policies don't need to be perfect right away.

STEP THREE: BUILD THE FOUNDATION

Without accountability, enforcement and automation, any policy blueprint is destined to fail.

All of the policies documented by the blueprint should be stored in a Repository accessible to all relevant constituencies.

The next step is to move from describing and documenting policies to establishing accountability, enforcement and automation of those policies. These require mechanisms for identity and security. Whether controlling access to a Repository asset at Design Time or enforcing a Service Level Agreement (SLA) at Run Time, identity and security are fundamental to those Policy Enforcement Points (PEPs).

Once identity and security are established, the other policies defined in the Blueprint become enforceable. The nature of enforceable policies is a function of the governance system you have in place. This paper will describe the set of policies which can be described and enforced by the X-Registry product.

When the responsibilities of service providers are enforced by policy infrastructure, service consumers can consume services with trust and confidence and understand that identified providers will be accountable to the impact of the behavior of their service against commonly shared expectations. Well defined policies can also help ensure that unintended consequences in your SOA implementation don't result in disruption of service or a breach of business or regulatory requirements.

In some respects, Enterprise SOA more closely resembles the design of a town, a city or even a country. A well-designed SOA balances the need for infrastructural services with the need for commercial services—much like a mass-transit system or municipal water system needs to be scalable to support the needs of a town as it grows. This suggests that infrastructure should not dramatically exceed the needs of the supported community, but that careful planning can ensure that infrastructure can grow alongside the applications.

When SOA policy enforcement reaches its most mature state, it can resemble the balance of powers seen in a Federal government model, where the policies and jurisdiction of distributed states must be reconciled with the policies and jurisdiction of the central government. This resembles the federated policy model established between central IT and distributed business units.

SOA GOVERNANCE FOUNDATION REQUIREMENTS

"Companies moving to implement Enterprise-scale SOA need Registry/Repository and governance capabilities… 2006 should prove to be an even more significant year for Registry/Repository providers."

Ron Schmelzer, Senior Analyst, Zapthink

As mentioned earlier, SOA Governance cannot be purchased in a packaged product. However, the foundation of SOA Governance is the ability to enforce and automate policies across the lifecycle.

Therefore a set of mechanisms can enable the automation and enforcement of SOA policies. This section will outline the typical components of an SOA system which perform these policy-related functions.

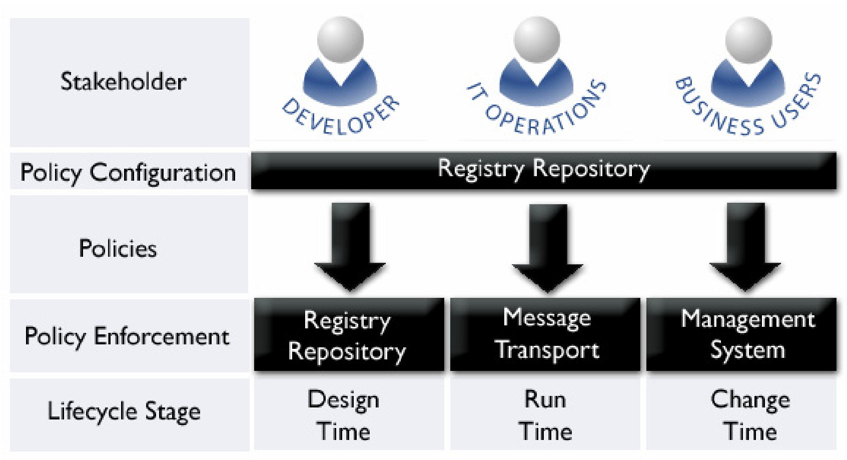
To do this, you need a place to store policies (Policy Repository), and a place to enforce policies (Policy Enforcement Points). These two locations represent where policy "lives" and where policies "go to work."

The SOA lifecycle consists of three stages as follows:

1. Design Time
2. Run Time
3. Change Time

Regardless of lifecycle stage, the Registry/Repository is the Policy Store for all stages in the lifecycle. Having a single (federated) context for policies enables lifecycle management of policies and a means to establish system-wide policies.

However the Policy Enforcement Point (PEP) changes, depending on the lifecycle stage. As shown in the diagram below, during Design Time, the Registry/Repository itself is the PEP. During Run Time, the Message Transport system is the PEP. Finally, during Change Time, the Management and Security system is the PEP.



Reader's Note:

Many of the Lifecycle Governance policies in Design Time apply to services regardless of whether they are Web Services or Non-Web Services. However, enforcement of Run Time policies for Non-Web Services depends on the Message Transport system and the application.

Policy Management and Enforcement is described in more detail, below:

The Policy Repository ensures that all policies are governed, versioned, approved and managed across the lifecycle of policy enforcement.

During the Design Time of SOA, policies such as namespace validation, schema validation, interoperability validation, lifecycle approvals, document access control, auditing and resource utilization need to be enforced.

If the Policy Repository is combined with a Service Registry, then the typical best practice is for the SOA Registry/Repository system to also serve as an enforcement point for these types of policies, thus the Registry/Repository acts as both the Policy Store across the lifecycle, and a Policy Enforcement Point for Design Time.

The Run Time system is responsible for Message Transport. It is typical to use systems such as an Enterprise Service Bus (ESB). These systems broker transactions between service provider and service consumer and frequently handle functions such as data transformation, reliable messaging and message queuing, security, and other operational concerns. In Web services environments, the emphasis is on supporting SOAP and extensions such as WS-Security. These systems can act as Run Time Policy Enforcement Points (PEPs).

A specialized class of hardware appliance, referred to as an XML Gateway or SOAP Firewall or other names, is rapidly becoming a popular specialized component in the policy governance system during Run Time. Although these devices are Message Transport systems, unlike mainstream ESB components, these appliances are frequently deployed as security and XML acceleration devices, most often at the “edge of the network,” between organizations or between companies.

Change-time governance is the act of managing services through the cycle of change. Since most services will be modified many times, this is an essential component of long-term governance. During change time, users who are connected to services, taxonomies, or policies receive notifications to approve, review, or recognize when any of those assets are modified. This oversight becomes even more critical when managing multiple versions. Furthermore, change-time enforcement enables policies to dictate if a service or even another policy can change and, if so, who needs to approve the change. If the change is planned (i.e. version upgrades, operational changes, service decommissioning, service deprecation) then it should be recorded, and described (semantically) to allow the dependent parties to adjust their processing.

PART ONE CONCLUSION: SOA LIFECYCLE GOVERNANCE REQUIREMENTS

“A metadata repository is a key enabling technology for SOA. It is safe to say that no long-term enterprise SOA initiative can succeed without an integrated and searchable repository/registry.”

Yefim Natis, VP and distinguished Analyst, Gartner

SOA provides organizations with a powerful opportunity to transform the way in which they deploy Information Technology for strategic advantage.

These can include benefits to both central IT and to business units. These combined benefits offer a unique opportunity to bridge business and IT and create previously unattainable alignment between these organizations.

The benefits can include consolidation of resources and elimination of redundant functionality through reuse, and increase visibility, control and compliance for Central IT. For business units, SOA holds the promise of increasing the agility and flexibility of IT resources, particularly improving the visibility and adaptability of business processes.

However, this opportunity is constrained by a set of challenges which can be helped by developing SOA Governance expertise, sets of policies and ways to enforce those policies across the SOA Lifecycle.

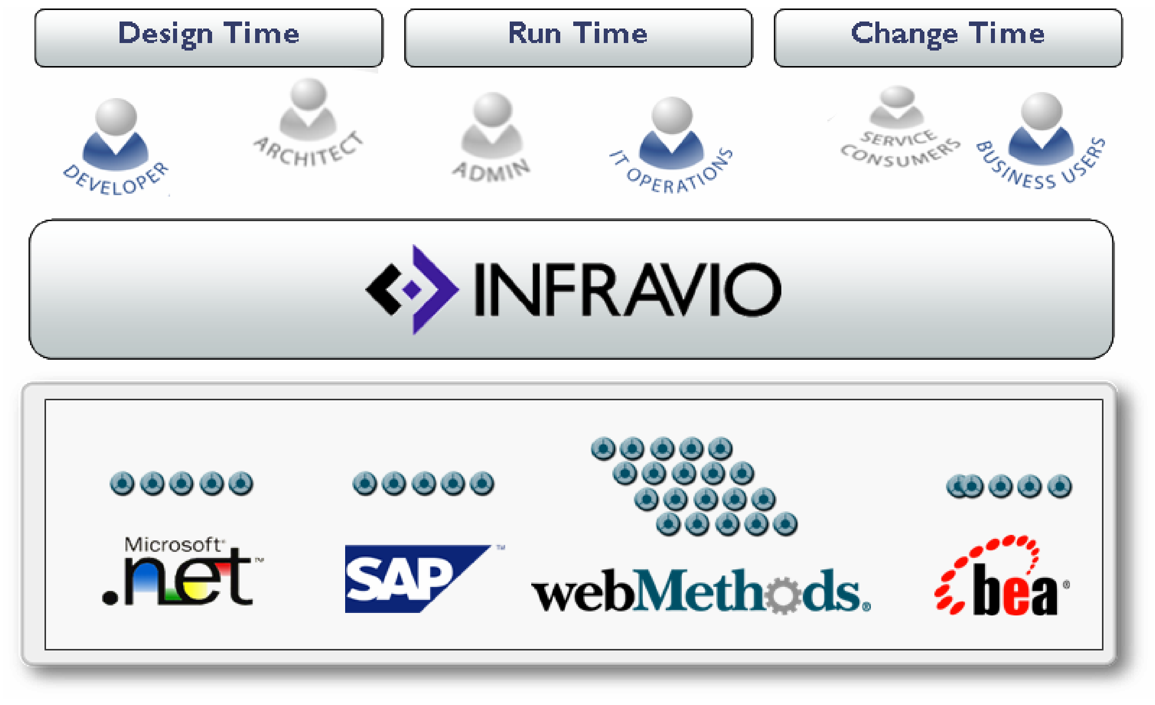
The foundation for SOA Lifecycle Governance are Registry/Repository, Message Transport, and System Management components for enabling multiple stakeholder organizations to collaborate, design, store and enforce Governance policies. These systems should be interoperable with one another to ensure that policies defined in one system are enforced in the others.

PART TWO: INTRODUCING X-REGISTRY

SOA GOVERNANCE AND LIFECYCLE MANAGEMENT

The X-Registry product is a software platform for SOA Governance and Lifecycle Management. At its core is a Registry/Repository, which serves as a service catalog and repository for policy and related artifacts.

The X-Registry is compatible through SOA Link (described later in this white paper) with most Message Transport systems including ESBs from top vendors. X-Registry also achieves compatibility with most management systems through SOA Link.



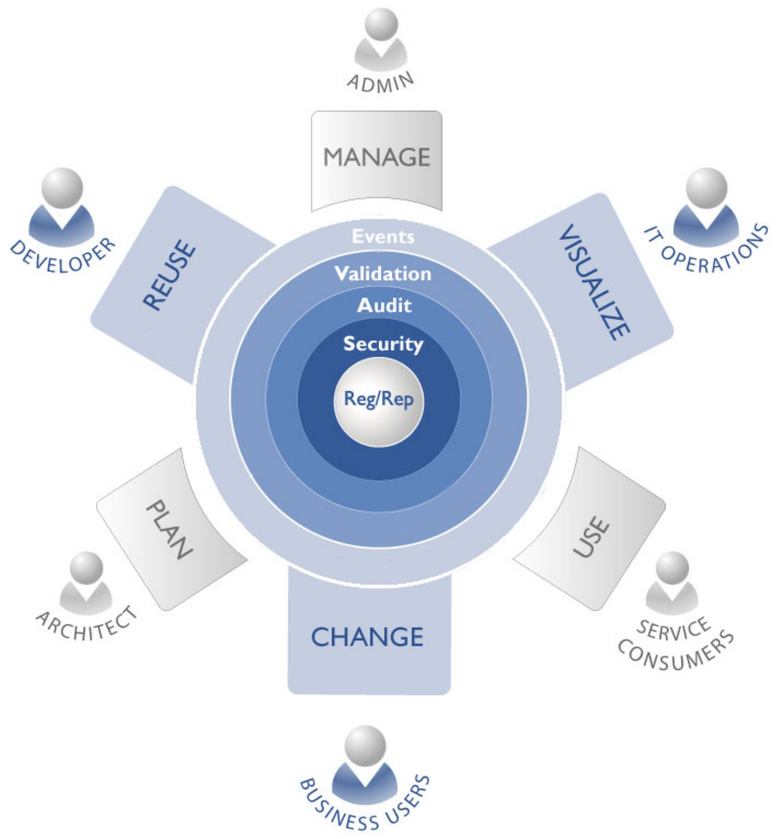
webMethods provides a compatible Message Transport system as part of the X-Registry product called X-Broker. This provides rich run-time functionality for Web service mediation, but is not a full Enterprise Service Bus. Complete ESB functionality requires several webMethods products including Integration Server, Broker, Infravio X-Registry, and Infravio X-Broker.

The most significant challenge indicated by the OASIS SOA Reference Model definition is that distributed capabilities *may be under the control of different ownership domains*.

This makes SOA a profoundly multi-user system. Much like computer operating systems help applications schedule and share resources (e.g. CPU, disk and memory) SOA needs a mechanism that enables, supports and schedules shared services and resources across a network. X-Registry is just such a system.

The diagram below shows how multiple stakeholders across the service lifecycle all align and interact with the same Registry/Repository. Although each stakeholder is responsible for a different stage in this lifecycle, the product helps align all these players and ensure the reliability, predictability and quality of your services.

Each of the layers in the diagram below denotes an aspect of Registry/Repository policy which is enforced within the product. X-Registry also enforces Run Time policies through integration with popular ESB, Web Services and Messaging products.



X-Registry serves as a Governance system of record which applies to both design time and run time SOA policies.

Below, this white paper provides details as to the types of policies the X-Registry can enforce during Design Time, Run Time and Change Time. The types of policies that can be enforced determine the completeness of a governance foundation throughout the SOA Lifecycle.

Reader's Note:

Design Time Governance typically concerns assets checked in to the Registry/Repository

Run Time Governance typically concerns the messages flowing across the Message Transport system.

Understanding this distinction is the key to understanding SOA Lifecycle Governance.

The Policy Enforcement Point (PEP) for SOA Design Time is the Registry/Repository.

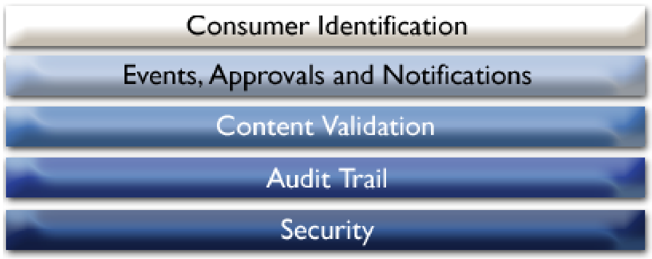
Since the Registry/Repository is the system of record for both service interfaces as well as all the assets, attributes and metadata associated with them, it's a logical point at which to enforce policies about the design of your SOA.

Design Time policies are typically applied on artifacts that are checked in to the Registry/Repository including (non-exhaustive list):

- WSDL
- XSD
- Service Documentation
- Schemas
- Name Spaces
- Taxonomies
- WS-BPEL
- Any documents checked in to the Registry/Repository

For example, a service consumer expects interfaces to be well-formed and interoperable, so the Registry/Repository should automate the process of scanning and assuring that WSDL documents well-formed and standards compliant. Similar scans can check namespaces and schemas. The Registry/Repository can also support fine-grained access controls and federated lifecycle promotion, for example migrating services and appropriate metadata from a development instance to a staging server. Thus the Registry/Repository can act as a design time policy enforcement point to ensure that the SOA is meeting the guidelines set forth by architects and planners.

During the SOA design and assembly stages, the Registry/Repository itself serves as a Policy Enforcement Point (PEP) for a set of polices which help to ensure a well-formed SOA project. These policies are listed below and in the following diagram.



Each of the policy areas is described in greater detail, below.

X-Registry supports a number of Registry/Repository user identification mechanisms. Principally, users can be identified through integration with LDAP based identity systems including Microsoft Active Directory.

Consumer identification is the first step in establishing rights and responsibilities in the Registry/Repository. While the product supports a “guest” user, typical interactions with the Registry/Repository are with identified users and is therefore metered, audit logged and subjected to approvals and appropriate governance processes.

Any interactions with the Registry/Repository involving Create, Read, Update or Delete events can be configured to trigger automated responses.

These automated responses include alerts, approval processes, content validation scans or a variety of metadata configuration based changes. These triggers can be applied either before or after the interaction in question. For example, a policy can be established that approval is needed *before* an object is created in the Registry/Repository. This means that prior to the approval, any object submitted for insertion will not exist in the Registry/Repository. If the policy applied is automatic, such as a content validation scan, it can automatically accept or reject such events and interactions.

Content is initially validated by a set of content types in the Registry/Repository. For example, a date field must reflect a legitimate date, similarly, data such as duration cannot be a negative value. All of these typed validations are automatically reflected in policies that ship with the product.

Beyond this, artifacts can be scanned for any detectable policy enforcement event. Common validations include testing for namespace violations, schema validation, WSDL validation and other interoperability related scans.

A fundamental capability for establishing accountability in SOA is the ability to establish exactly the actors, sequence and details of every interaction with the Registry/Repository. This record can be used for governance enforcement “after the fact,” it can be used to establish usage patterns, can be a way to reverse events or reliably maintain state across a federated set of registries in situations when they are disconnected.

Another essential set of policies is the access control system. The X-Registry Access Control system is the most powerful SOA Governance Access Control solution in the market. This system allows for fine-grained access configurations on all aspects of Registry/Repository assets. This includes the ability to secure assets, but also metadata, policies, processes, classifications, and documents.

This access control system is supported by a delegated authority organizational model. This enables a powerful organizational set of models which enables provider and consumer organizations to coexist in a multi-tenant federated Registry/Repository context.

The Policy Enforcement Point (PEP) for SOA Run Time is the Message Transport system.

Message Transport is carried out in SOA by a number of products, notably Enterprise Service Bus (ESB), Web service brokers, XML gateways, integration hubs, legacy adaptors and custom-built intermediaries.

Run Time policies are typically applied on messages that flow across the Message Transport system including (non-exhaustive list):

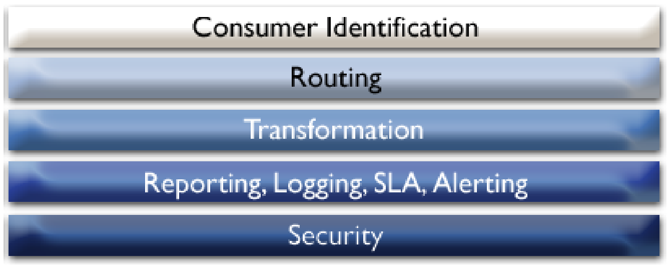
- SOAP Messages
- WS-Security
- XML Encryption
- XML Signatures
- XSL Transformations
- Authentication events
- Transactions
- Any messages flowing across the Message Transport system

Since SOA events, documents and transactions all flow through the Message Transport system, it's a logical point at which to enforce policies during Run Time of SOA.

For example, one set of service consumers might be accessing a service from outside the corporate firewall. This consumer might then fall under a policy assertion that all traffic leaving the company be secured by 1024 bit encryption keys. In the case of specifically identified consumers, other policies may be applied such as a preference for X.509 certificates or for the delivery of low latency service levels such as a 400msec SLA.

During the SOA Run Time stage, the Message Transport system serves as a Policy Enforcement Point (PEP). This is often an Enterprise Service Bus (ESB) but can also be any of a host of other integration hubs, adaptors, proxy servers, portals, web service fabrics, SOAP interceptors, custom built brokers, XML gateways/firewalls and other such products.

Supported Run Time policies are listed below and in the following diagram.



Each of the policy areas is described in greater detail, below.

As the first security step, the X-Registry identifies the consumer application to prevent unauthorized applications from accessing the Web Service. The identification criteria can include the following:

- Request header patterns
- User group as identified by the security
- Authentication server
- IP address range
- Digital certificate

X-Registry allows the consuming application to specify the security method that best meets their needs. End user authentication and authorization can be done by invoking 3rd party systems. The product currently supports:

- LDAP server
- Microsoft Active Directory

Service consumer identification is an essential first step to establishing policy enforcement at Run Time.

X-Registry supports a substantial number of dynamic routing policy configurations including failover, and clustering strategies including blind, round robin and least busy. It also supports dynamic routing used for version management, deprecation, custody transfer and other service lifecycle issues.

The policies that can be configured using X-Registry include:

- Content based routing—A SOAP message can be examined to validate a specific business rule. Based on the rule, the message can be routed to different back-end services.
- Preferential QoS—Based on business priorities and defined service levels, consuming applications can be assigned different processing priority. By defining QoS business rules, specific applications can receive expedited response, lower priority response, or even the initiation of a different service version.

- Version based routing—X-Registry can support policies that enforce automatic routing to different versions of a service for a consuming application. If multiple service versions are available, the broker can determine the correct version

As with all Run Time policy configuration, these can be established through the X-Registry web-based interface with reusable Service Delivery Contracts™ and policy templates.

webMethods Integration Server can be configured to work with Infravio X-Registry to enforce policies that facilitate data transformation and application connectivity. These functions include:

- Transport Brokering—X-Registry supports a variety of transports and can effectively broker between different transports on the provider side (Broker to Service) and on the consumer side (Client to Broker). Supports both MQ and JMS.
- Transformation—X-Registry can leverage an XSL Transformation engine as specified in the Service Delivery Contract™. X-Registry can also invoke 3rd party transformation engines if necessary.

.

Depending on the enforcement points and use cases, these application integration features can represent a significant part of the requirements for mediation in SOA.

X-Registry includes a console for performance, availability management and activity monitoring of services.

The Message Transport system can be configured to capture a rich set of data based on the rules established within the contract. It streams that data to a “console server” an integral part of the X-Registry, which acts as the aggregation point for all of the data from the distributed broker network.

The collection of data by the Message Transport system does not impact service performance. The following functions are performed:

The logging and monitoring term allows rules definition for logging, monitoring and alerts based on service parameters. Service level data is collected at Run Time by the broker and is available at two levels.

- Transaction level data includes information about every request/response message sequence relevant to the consuming service, service provider and the contract that governs their relationship.
- Aggregate data is relevant to the overall performance of the service. For example, logging and monitoring rules can be established based on aggregate counters for performance, errors, response times, and SLA violation events.

The X-Registry can automatically route a request to a backup service in case of failure in the primary service. An easy-to-use template is available to configure ‘fail-over’ routing.

- . Performance Management—Routing can also be used to equalize or prioritize the request load among multiple backend services. Easy to use templates are available to configure this ‘load balancing’ capability.
- SLA Management—At Run Time, the X-Registry can enforce priorities to match the response times of an SLA and send alerts if the SLA is not met.

The console server exposes the management information and functions available on each agent/broker in compliance with the OASIS Web.

The X-Registry offers a wide range of features to configure the security of services at run time and is fully compliant with all WS-Security specifications.

The mediation system is flexible, can adapt to the specific needs of different consuming applications, and facilitates the implementation of a Service Oriented Architecture. It enforces the Service Delivery Contract™ security terms agreed between a specific consumer application and the provider.

In addition to consumer application and end user authentication and authorization, the X-Registry offers the following Run Time security features:

- Encryption and decryption for message confidentiality
- Digital signatures for message integrity and non-repudiation
- Security alerts to prevent hacking
- Logging for tracing and tracking, IP blocking to counter denial of service attacks

Change Time is a unique lifecycle stage in SOA. Older software paradigms featured “software maintenance” and an expensive and inflexible state of software once deployed. Because SOA is capable of metadata-based, policy-based and process-based changes, it has the potential to be in a continuous state of optimization and adaptation.

Registry/Repository is the system of management for policies across the SOA lifecycle, therefore it contains the metadata patterns important for Change Time. Change time service governance includes:

- Service change subscription
- Service binding subscription
- Service metadata subscription
- Email change notification
- SOAP notification
- Synchronous/asynchronous notification
- Service relationship and dependency management
- Impact analysis
- Change Time version management (depreciation, migration, expire, etc.)

Since SOA events, documents and transactions are all enforced by X-Broker during the run time and monitored by Business Activity Monitoring tools such as webMethods Optimize during change time, it's a logical point at which to bring Run Time information back into the Registry/Repository. This type of information is essential to optimizing service delivery during Change Time of SOA.

webMethods Optimize can bring key information into the Registry/Repository including:

- Key Performance Indicators (KPI)
- Business Activity Monitoring (BAM)
- Service Level Agreement (SLA)
- Business Process Visibility

This type of information can drive the change process, which can involve adjustments in metadata and policies, roles and processes, or in the services themselves.

X-REGISTRY PLATFORM 6 CORE

At the core, the product contains an integrated Registry/Repository, SOA Information Model, and Governance Rules Engine. X-Registry was the first product in the industry to integrate standards-based Registry and Repository in a single product.

Building Registry and Repository as two separate products creates significant potential dangers as there can be de-synchronization of the content in both locations as well as an inconsistent information model. This can be seen by requirements to enter redundant data in two locations or products. By creating a true unification of Registry and Repository into one product with one normalized and standards-based Information Model, webMethods is able to ensure the integrity of both data and data model.

The first step in achieving reusable services for SOA is cataloging existing services. OASIS provides a Web Service standard for registry interoperability known as UDDI. UDDI provides mechanisms for service discovery, including hierarchical classification schemes (taxonomies) and searchable attributes. The role of the registry in Web Services is to serve as a service catalog, or "system of record" for all the other services within an SOA.

The X-Registry supports the full set of UDDI version 2 and UDDI version 3 Application Programming Interfaces (APIs). This enables integration with all popular Integrated Development Environments (IDEs) and also serves as an integration point with run time Message Transport systems and Management systems.

Registries (or directories) form a key component of distributed systems architecture—a registry is a unique service within SOA because its function is to store a record of all of the other services. Having a catalog or index of services is significant from an organizational perspective, because it assures the easy discovery and reuse of services. As the only service whose job it is to "know about every other service," it is a natural management point for your services—commercially available Registries often have robust interfaces for management and administration of your services. A Registry is by definition a lightweight construct. A Registry is not designed to store assets, but to indicate their location by reference. Registries provide federation capabilities as well as classification to ensure the ability to index a widely distributed set of assets.

Increasingly organizations understand the need for a policy repository. Unlike a Registry, Repositories store (rather than indicate with pointers) assets. Repositories are heterogeneous and distributed in nature.

However, the requirement to codify and enforce organization and regulatory policy usually calls for a centrally managed policy repository.

The X-Registry is logically centralized, but topologically distributed. This results in a federated architecture. This prevents bottlenecks and fiefdoms from arising—but logical centralization emerges from the need for a single "official" set of policies which can be consulted, adapted and enforced. Policies consist of sets of human and machine readable (and in some cases readable by both) documents which describe rules that govern the appropriate constraints within which business processes are executed. In some cases, policies are enforced by humans, in others they are automatically enforced by machines, such as Policy Enforcement Points.

X-Registry provides support for federating multiple instances of Registry/Repository across organizations.

Federation may be a requirement of supporting multiple geographies or multiple lifecycle constituencies. In some cases, the federation of instances may occur across company boundaries.

X-Registry supports the ability to ensure that all interactions with any Registry/Repository instance will appropriately federate with all other instances in that context. Appropriate federation is established through sets of policies that determine the appropriate information flow between instances, whether that be automatic, or triggered by events such as service lifecycle promotion.

X-Registry is built on an extensible standards-based information model. It supports the UDDI interface for Registry, the ebXML Registry/Repository interfaces for Repository, and the JAXR (Java API for XML Registries), which reconcile and integrate the two.

The ebXML Registry/Repository specifications are standards under OASIS, the United Nations United Nations Centre for Trade Facilitation and Electronic Business (UN CEFACT), and also are standards under the International Standards Organization (ISO) under ISO15000-3.

The base object is the Registry Object from which all entities in the Registry/Repository derive. Similar to the ebXML Registry/Repository Information Model (ebRIM), X-Registry provides an extensible attribute model that enables extension of the metadata on an object. Other fundamental primitives include links to external objects, associations between objects, classifications and taxonomies, auditable events, users, organizations, services, and service bindings.

The Information Model is extensible in a standards-based way through the user interface. Extended attributes are fully typed, validated and searchable through the user interface, and are automatically integrated throughout the product. Taxonomies are similarly extended and integrated.

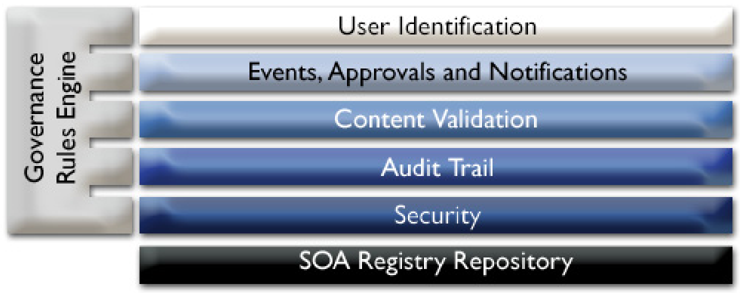
This information model enables the control of the underlying Registry/Repository by the user interfaces, but also by a set of programmatic interfaces including:

- The Governance Rules Engine
- Governance API
- Java API for XML Registries (JAXR)

The Governance Rules Engine can be programmed declaratively through sets of XML rule documents, through Java or additional scripting languages such as Groovy.

This Information model is compatible and can be accessed through the UDDI standard via tModels and CategoryBags and also the ebXML Registry/Repository specifications.

Another unique aspect of the X-Registry is that it is the first and only product in its class to embed Rules Engine technology for automating SOA Governance functions. All of the layers of policies enforced by the Registry/Repository can be automated by this rules engine.



The rules engine integrated with X-Registry is a full Java JSR-94 compliant Java Rules Engine, comparable with commercial offerings from iLOG or Fair Issacs.

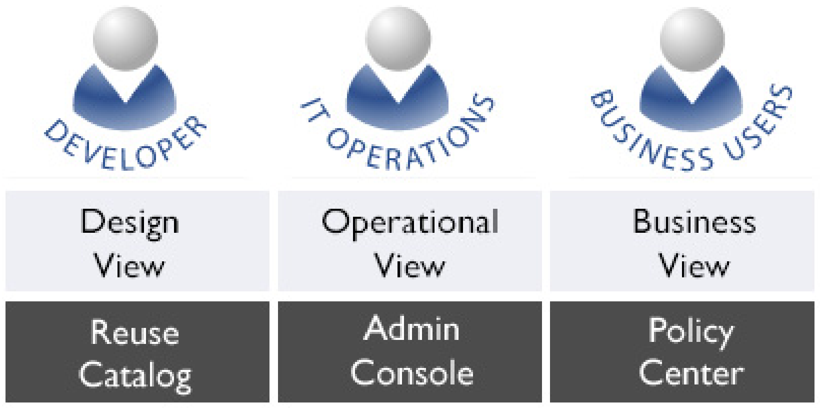
This enables the automation of complex governance tasks such as setting and resetting access control switches at lifecycle milestones such as promoting a service from development into staging or production.

In addition to automating governance tasks, the governance rules engine can also help to deal with policy federation. This means that multiple policy authors and authorities can establish guidelines for how to reconcile policies that come into conflict. This is an important use case in Enterprise scale SOA when a single organization or individual does not have full control over all SOA policies.

Governance products without Rules Engine capability are unable to implement policy federation. This forces all policies to be authored and controlled by a central organization—which defeats the concept of federation and distributed control. The rules engine also provides the most powerful and yet easy to use interface to creating complex policies based on reusable templates. This enables a broader set of users access to policy authoring.

A sign of an immature policy environment is one in which all policies are authored by the service providers. A more robust model is to enable both centralized policy creation as well as distributed. This is the basis for the federation of policy.

X-REGISTRY COMPONENTS



"One of the hallmarks of a Service Oriented Architecture is the large amount of associated documentation and description. The service description represents the information needed in order to use a service. In most cases, there is no one "right" description but rather the elements of description required depend on the context and the needs of the parties using the associated entity."

OASIS SOA Reference Model 1.0 Committee Draft

X-Registry supports service reuse through service artifact validation, interoperability testing, enhanced classification and discovery, demo-able services, extended service descriptions and service documentation.

Service "Profiles" that ship with X-Registry enable fine-grained specification of services based on SOA Governance Best Practices. The profiles that ship with the product include:

- Basic Profile (version, status, creation/expiration, URL, etc)
- Extended Profile (keywords, release date, etc)
- Classification Profile (taxonomies and attributes)
- Resource Profile (documentation and artifacts)
- Dependency Profile (dependencies and similar services)
- Performance Profile (service performance info)

These tools for service description are augmented by powerful classification tools including taxonomy import and export, taxonomy editor, dynamic attribute modeling, and advanced search.

In addition to providing powerful tools for service description, X-Registry provides support for powerful demo service capability which can help developers of consuming applications test service consumption without affecting the deployed service.

This application provides two functions to the Run Time operations of your SOA. Firstly, the Administration Console provides a dashboard view that allows support for an SOA dashboard to provide key metrics and reports of services, performance and other analytics. The second function enables the management of roles, organizations and permissions associated with the Registry/Repository.

The administration of X-Registry is strongly role-based and includes the following types of users:

Superadmin—Global access to organizations and SOA Assets

Provider Organization Administrator (POA)—Local administrator for an organization. Can manage users, groups, roles, services, access control, and policies.

Provider (PO)—A user that has privileges to publish information to the catalogue.

Consumer Organization Administrator (COA)—Local administrator for a consumer organization. Controls users and groups within the consumer organization, and has read-only access to SOA Assets.

Consumer (CO)—A read-only User that can access SOA assets. Consumers can search, browse, read and download.

Guest—A limited access, un-authenticated user that can read limited information, but can not download the following:

- Nodes
- Global
- Account
- Availability
- Notification
- Inquiry
- Publish
- Security
- Subscription
- UDDI Policy

The Policy Center application provides dependency and impact analysis and configurable Service Delivery Contracts™, as well as detailed configuration of Change Time policy automation.

The Policy Center provides a rich web based user interface that enables the configuration of SOA policies from design time through run time and change time. Configuration of policies are implemented through fully reusable policy templates.

Run Time policies are configured using the webMethods Infravio Patent-Pending Service Delivery Contract™ technology. The robust automation of policy configuration through change time enables businesses to make changes in the configuration of SOA which can transform the behavior of fully deployed services. This provides an extraordinary degree of agility as it can dramatically reduce the need for developing custom versions of each service due to service customization needs.

INFRAVIO SERVICE DELIVERY CONTRACTS™

Service Contracts enable organizations to configure attributes that govern the relationship between service providers and consumers, enforces service policies across the Lifecycle, facilitates services reuse, and increases business agility as organizations move toward a Service Oriented Architecture (SOA).

Our patent pending implementation, the Service Delivery Contract™ (SDC), allows SOA builders to create generic services and then define delivery terms such as security, routing, load balancing, failover, Service Level Agreements (SLA), alerting and notification, data transformation or monitoring that are specific to the consuming application. The result is a dynamically configurable set of services, preferential treatment of service consumers, and faster integration and Change Time Governance capabilities for your SOA.

Preferential treatment is the hallmark of a mature SOA. Not all service consumers should be treated alike. The ability to easily configure, govern, and track interactions between providers and consumers in the end-to-end service lifecycle enables agile delivery of business services.

By maintaining delivery preferences in a contract, loose coupling and rapidly adaptable configurations between service consumers and services is achieved.

Policies and contracts metadata are externalized in the SOA Repository. These are documents which in some cases are human readable and in some cases machine readable. Service Delivery Contracts™ makes contract documents readable by both humans and machines. In the case of Service Delivery Contracts™, business users can easily manage and make configuration changes to deployed services through a browser based interface.

It is a best practice to model the fastest changing aspects of your business in metadata. This makes changing in response to business requirements faster and easier. One way to think about this layer is that how all of these aspects reflect unique capabilities, limitations and preferences of the service consumer.

SOA LINK

The webMethods Infravio X-Registry ensured Governance Interoperability through membership in the SOA Link organization.

SOA Link is an end-to-end SOA Governance interoperability initiative. Organized by Infravio, Inc. prior to its acquisition by webMethods, SOA Link brings together multiple vendors with products for SOA Governance to mutually interoperate. Whether supplying a policy Repository and authoring system, Run Time enforcement system, monitoring Run Time testing and quality assurance, or business process capability, these systems will be delivered as interoperable by leading vendors and technologies. Participants benefit by having the ability to publish services and associated policy to a system of record in a standardized way, and to be alerted to changes when they happen. SOA Link members include AmberPoint, Composite Software, Forum Systems, HP, Intalio, IONA, iTKO, JBoss, Layer 7 Technologies, LogicBlaze, Mindreef, NetIQ, Parasoft, Reactivity, SOA Software, Solstice Software, SymphonySoft, and webMethods. Their logos are presented below. SOA Link is the largest Governance Interoperability organization in the market.

SOA Link is not a standards body; the primary purpose of SOA Link is to support Governance Interoperability through the publication of the SOA Link Catalog, a web site found at http://www.soalink.com.

In addition, vendors participating in SOA Link are assumed to support the following principles:

Whenever possible, SOA Link members will integrate their solutions using available standards. The SOA Link organization recognizes the work of standards organizations such as W3C, OASIS, ISO, IETF, ECMA and many other groups. Many of the member companies are represented in these standards groups. SOA Link also recognizes the work of WS-I, the Web Services Interoperability organization.

By joining the organization, the vendors express a desire and willingness to interoperate with other solutions in the SOA Link catalog, and with other SOA Link member organizations.

Members support the open publication of use cases, specifications, integrations, announcements, partnerships and other important information regarding Governance Interoperability to the SOA Link web site. Each member organization will designate a representative to maintain their vendor “channel” on the site.

SOA Link use cases and integrations will take a customer centric perspective, and will partner with end-users of technology to understand the requirements for integration.

PART TWO CONCLUSION: X-REGISTRY

Enterprises seeking the benefits of SOA should consider applying governance to ensure positive outcomes. This white paper describes an approach to SOA Governance which establishes a catalog of existing service assets, creates a policy plan for establishing mutual accountability and recommends a foundation set of products for ensuring these policies are implemented throughout the SOA Lifecycle.

This white paper also describes the X-Registry product which serves as a cross-lifecycle, cross-organizational foundation for SOA Governance. The X-Registry represents over 150 person-years of engineering and product development work, and is the most mature and most complete SOA governance and lifecycle management platform available today.

This award-winning product represents the state-of-the-art in governance policy definition, enforcement and reporting. Along with SOA Link Governance Interoperability partners, X-Registry customers can be assured of robust governance which helps guide organizations to desired SOA outcomes.

ABOUT WEBMETHODS, INC.

webMethods provides business process integration to the world's largest corporations and government agencies. webMethods' flagship product suite, webMethods Fabric, is the only integrated platform to deliver both SOA and BPM, delivering rapid ROI to our 1,500 customers around the globe. With webMethods, customers can take a process-centric approach to their business problems, allowing them to leverage their existing IT assets, dramatically improve business process productivity and ROI, and rapidly create competitive advantage by making their business processes work harder for their company.

webMethods (NASDAQ: WEBM) is headquartered in Fairfax, VA, and has offices throughout the United States, Europe, Asia Pacific and Japan.

Worldwide Headquarters
3877 Fairfax Ridge Road
South Tower
Fairfax, VA 22030
USA
Tel: 703 460 2500
Fax: 703 460 2599

US West Coast
1198 E Arques Avenue
Sunnyvale, CA 94085
USA
Tel: 408 962 5000
Fax: 408 962 5329

European Headquarters
Barons Court
22 The Avenue
Egham, Surrey
TW20 9AB
United Kingdom
Tel: 44 0 1784 221700
Fax: 44 0 1784 221701

Asia-Pacific Headquarters
6 Temasek Boulevard, #24-01
Suntec Tower Four
Singapore
038986
Tel: +65 6389 3200
Fax: +65 6389 3299

www.webMethods.com

The webMethods logo and Get There Faster are trademarks or registered trademarks of webMethods, Inc. All other names mentioned are trademarks, registered trademarks or service marks of their respective companies.